Javascript avancé
Module 2 – Javascript orienté objet

OBJECTIFS

Connaitre le fonctionnement des objets en Javascript

Appréhender le principe de prototype

Utiliser la syntaxe ES6

SOMMAIRE

Programmation orienté objet

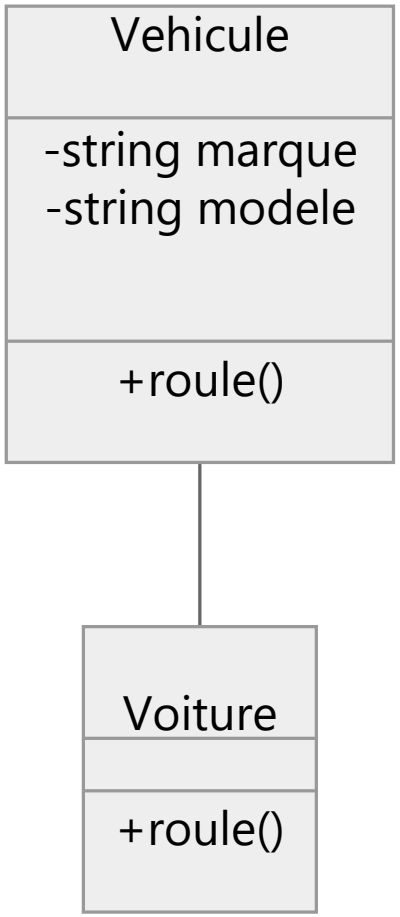
POO

- Paradigme basé sur le concept objet
- Un objet permet de décrire le monde réel
- Un objet contient des données (ou propriétés) et du code (ou méthodes)
- Un objet peut interagir avec un autre objet
- La POO permet d'organiser le code afin qu'il soit facile à maintenir

Le développeur tente de s'éloigner le plus possible du code spaghetti 

Principes

- Abstraction
- Encapsulation
- Héritage
- Polymorphisme



Javascript orienté objet

Les caractéristique d'un objet, en Javascript, sont :

- Un ensemble de clés/valeurs stocké en mémoire
- Des attributs facile d'accès
- Utile pour travailler en JSON (module03)

Tout ce qui n'est pas primitif est objet

Propriétés

```
let etudiant = {
    prenom: 'Chloe',
    administrateur: true,
    age: 29
}

console.log(etudiant);
```

```
console.log(etudiant.prenom); // Chloe
etudiant.prenom = 'Medhy';    // Modification
console.log(etudiant.prenom); // Medhy

etudiant['promotion'] = 'DWWM147';
console.log(etudiant); // {prenom: 'Medhy', administrateur: true, age: 29, promotion: 'DWWM147'}
```

Une propriété est accessible et modifiable

Méthodes

```
let sw1 = {
    titre: 'Un nouvel espoir',
    realisateur : 'George Lucas',
    annee: 1977,
    description: function () {
        return `"${this.titre}" a été réalisé par ${this.realisateur} en ${this.annee}.`;
    }
}

let sw2 = {
    titre: "L'empire contre attaque",
    realisateur : 'George Lucas',
    annee: 1980,
    description: function () {
        return `"${this.titre}" a été réalisé par ${this.realisateur} en ${this.annee}.`;
    }
}
```

⚠ La création de plusieurs instances d'un même objet entraîne une répétition de code

DÉMONSTRATION

Javascript orienté objet

Objets

L'objet ne va être défini qu'une seule fois :

```
function Film(titre, realisateur, annee) {
    this.titre = titre;
    this.realisateur = realisateur;
    this.annee = annee;
    this.description = function() {
        return `"${this.titre}" a été réalisé par ${this.realisateur} en ${this.annee}.`;
    }
}
```

Et les instance s'appuient sur cette définition :

```
let starwars1 = new Film('Un nouvel espoir', 'George Lucas', 1977);
let starwars2 = new Film("L'empire contre attaque", 'George Lucas', 1980);
console.log(sw1.description()); // "Un nouvel espoir" a été réalisé par George Lucas en 1977
```

DÉMONSTRATION

Fonctions

Prototypes

Prototypes

Les prototypes permettent à un objet d'hériter des propriétés d'autres objets

- Un objet est « lié » à un prototype
- Le prototype contient toutes les méthodes
- L'objet à accès à toutes les méthodes du prototype

Prototypes

```
function Film(titre, realisateur, annee) {
    this.titre = titre;
    this.realisateur = realisateur;
    this.annee = annee;
}

Film.prototype.description = function () {
    return `"${this.titre}" a été réalisé par ${this.realisateur} en ${this.annee}.`;
}

let sw1 = new Film('Un nouvel espoir', 'George Lucas', 1977);
```

Un prototype est attaché à un objet

Un objet est créé à partir d'un prototype

```
const prototypeFilm = {
    description: function () {
        return `"${this.titre}" a été réalisé par ${this.realisateur} en ${this.annee}.`;
    }
}

let sw2 = Object.create(prototypeFilm, {
    titre: {value: "L'empire contre attaque"},
    realisateur: {value: 'George Lucas'},
    annee: {value: 1980}
});
```

DÉMONSTRATION

Prototypes

ECMAscript2015 - ES6

Classes ES6

le mot clé class – un constructeur – des méthodes

```
class Film {
    constructor(titre, realisateur, annee) {
        this.titre = titre;
        this.realisateur = realisateur;
        this.annee = annee;
    }
    description() {
        return `"${this.titre}" a été réalisé par ${this.realisateur} en ${this.annee}.`;
    }
}

let sw1 = new Film('Un nouvel espoir', 'George Lucas', 1977);
```

i Tout ceci n'est que « sucre syntaxique ». En réalité, c'est toujours le mécanisme des prototypes qui est utilisé.

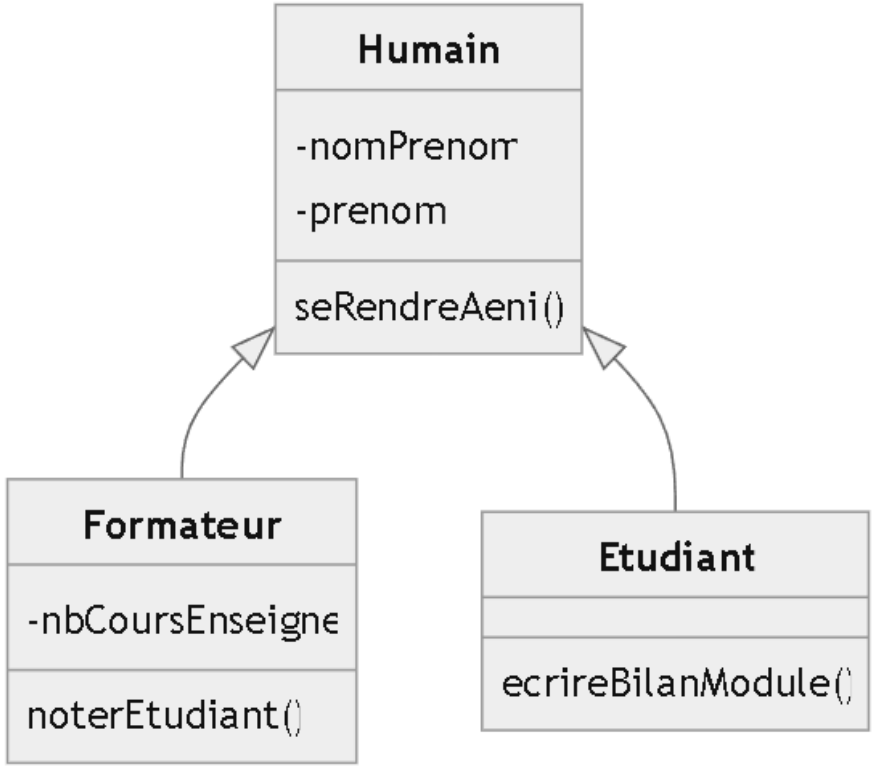
Propriétés

Par défaut, une propriété est publique.

Elle n'est privée que lorsqu'elle est précédée de « # »,

```
class Film {
    #titre; // Propriété privée
    annee;  // Propriété publique
    constructor(titre, annee) {
        this.#titre = titre;
        this.annee = annee;
    }
    get titre() {
        return this.#titre;
    }
}
let gladiator = new Film('Gladiator', 2000)
console.log(gladiator.annee); // 👌
console.log(gladiator.#titre); // ⛔
console.log(gladiator.titre); // 👌
```

Héritage ES6




```
class Humain {
    #nom;
    #prenom;
    constructor(nom, prenom) {
        this.#nom = nom;
        this.#prenom = prenom;
    }
    seRendreAeni() {
        console.log(
            "Je cours vers l'ENI " +
            "apprendre de nouvelles choses"
        );
    }
}
class Etudiant extends Humain {
    constructor(nom, prenom) {
        super(nom, prenom);
    }
    ecrireBilanModule() {
        console.log("Exceptionnel")
    }
    // [...]
}
```

Fonctionnalités ES6+

- Classes et héritages
- let et const
- Chaines de caractères
- Décomposition
- Déstructuration
- Fonctions fléchées
- Promesses / async / await
- Paramètres par défaut
- Spread
- Boucle for...of

DÉMONSTRATION

ES6

TRAVAUX PRATIQUES

Quizz

Conclusion

- Vous savez comment fonctionne les objets en Javascript
- Vous savez utiliser les prototypes
- Vous connaissez des éléments de la syntaxe ES6